

QUICK FACTS ON THE CAPABILITY MATURITY MODEL

The Capability Maturity Model for Software (CMM) is a framework that describes the key elements of an effective software process. The CMM describes an evolutionary improvement path from an ad hoc, immature process to a mature, disciplined process.

The Capability Maturity Model (CMM) is a robust model for improving the software process in any organization. When properly implemented, it effectively reduces and raises quality in a measurable way. The CMM has become the industry's leading process improvement model. The Software Engineering Institute has taken a lead role in producing research and refinements in the area of technology management. Refer to "The Capability Maturity Model, Guidelines for Improving the Software Process", Carnegie Mellon University, Software Engineering Institute, Addison-Wesley, 1995.

To help organizations and customers, the Software Engineering Institute (SEI) has developed the Capability Maturity Model for Software (CMM) that delineates the characteristics of a mature, capable software process. The progression from an immature, unrepeatable software process to a mature, well-managed software process also is described in terms of maturity levels in the model.

The Structure of the Capability Maturity Model (CMM)

- Level 1: Ad hoc
- Key Process Areas for Level 2: Repeatable
- Key Process Areas for Level 3: Defined
- Key Process Areas for Level 4: Managed
- Key Process Areas for Level 5: Optimizing

Brief overview of CMM Key Process Areas for Level 2 + Peer Review

1. Requirements Management

- Establish a common understanding with the customer.
- Requirements are clearly stated, verifiable, tracked, testable, controlled.
- Requirements and changes to the requirements are incorporated into model plans, products, and activities in an orderly manner.

2. Model Software Project Planning

- A statement of work is created that includes goals and constraints.
- A written organizational policy is followed; it includes steps to estimate the size of the model work products, resources needed, develop an activity schedule, identify assets and risks, and negotiate commitments.
- Estimates and plans are documented for use in tracking activities and commitments.

3. Model Software Project Tracking and Oversight

- Actual results (of costs, time, schedules, functionality) are tracked against plans, and reviewed by management.
 - Corrective action is taken for significant deviations.
 - Changes to commitments are understood and agreed to by all affected groups and individuals.
4. Model Software Subcontract Management
- Each project follows a written organizational policy for managing the model subcontract.
 - Select qualified model development subcontractors.
 - Establish documented commitments with each subcontractor that are understood and agreed to by both parties.
 - Track the subcontractor's actual results and performance against the commitments.
5. Model Software Quality Assurance
- Compliance of the model products and activities with applicable standards and procedures is independently confirmed, and follows a written organizational policy for implementing MQA.
 - Senior management is made aware of noncompliance issues that can't be resolved within the project, and addresses them.
6. Model Software Configuration Management
- Each project follows a written organizational policy for implementing model configuration management.
 - Selected work products are put under MCM control for baseline and changes, to maintain the integrity and traceability of the configuration throughout the life cycle, particularly for building the production model; work products could include model code, algorithms, data sets, requirements, designs, test data, procedures, calibration & verification results, sensitivity analysis, tools, etc.
 - Status and content of the model baselines are known.
7. Peer Reviews
- Model work products are reviewed by the producer's peers to identify defects and areas where changes are needed; this is done consistently using the methods for specific work products as defined in the standard development process.
 - Defects found are documented and tracked for removal.

The project plan is structured and organized in the following manner:

- Planning and Analysis:
 - Gap Analysis
 - Process Asset Library
 - Communication Plan

- Training Plan
 - Other planning activities
- Key Process Area (KPA) Development and Implementation:
 - Validate existing processes against baseline
 - Define KPA Goals
 - Prepare / Conduct Orientation
 - Define KPA Commitments and supporting Policies
 - Define KPA Activities
 - Design / Refine processes
 - Perform Activities using new/updated processes
 - Define Measurements, Analysis and Verification
 - Capture Measurement/Verification Data
 - Organize KPA Documentation / Assemble KPA Handbook
 - Update Process Asset Library
 - Identify and log KPA Artifacts
 - Conduct KPA training
 - Create and perform KPA checklist
 - Communicate progress / status update